UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/658,593	09/08/2003	Arndt Rosenthal	13913-136002 / 2003P00413	1930

22852 7590 02/09/2007

FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER LLP
901 NEW YORK AVENUE, NW
WASHINGTON, DC 20001-4413

EXAMINER	
YIGDALL, MICHAEL J	
ART UNIT	**PAPER NUMBER**
2192	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	02/09/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No.	Applicant(s)	
	10/658,593	ROSENTHAL ET AL.	
	Examiner	Art Unit	
	Michael J. Yigdall	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *08 September 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-19* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-19* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *26 May 2004* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-19 are pending. A priority date of September 8, 2003 is considered.

Specification

2. The use of the trademarks "DYNPRO" and "WEB DYNPRO" has been noted in this application (such as in claims 2, 9, 13 and 17, for example). They should be capitalized wherever they appear and be accompanied by the generic terminology.

Although the use of trademarks is permissible in patent applications, the proprietary nature of the marks should be respected and every effort made to prevent their use in any manner that might adversely affect their validity as trademarks.

Double Patenting

3. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory obviousness-type double patenting rejection is appropriate where the conflicting claims are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g., *In re Berg*, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent either is shown to be commonly owned with this application, or claims an invention made as a result of activities undertaken within the scope of a joint research agreement.

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

4. Claims 1, 3, 6, 7, 12, 14, 16 and 18 are provisionally rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claims 6, 10, 12 and 14 of copending Application No. 10/658,684.

Although the conflicting claims are not identical, they are not patentably distinct from each other because they recite analogous subject matter. As an example, the following table compares claim 1 of the present application to claim 6 of the copending application (note that claim 6 of the copending application incorporates claims 1 and 3 thereof):

Claim 1 (Present Application)	Claim 6 (Copending Application)
A computer program product, tangibly embodied in an information carrier, the computer program product being operable to cause data processing apparatus to perform operations comprising:	A computer program product, tangibly embodied in an information carrier, the computer program product being operable to cause data processing apparatus to perform operations comprising:

receiving run-time code for an application, the run-time code being generated from a converted design-time representation of the application, wherein:

the converted design-time representation of the application is generated from an original design-time representation of the application developed for use in a first run-time environment for executing applications having been developed in a first design-time environment, the first design-time environment using a first programming model comprising one or more first model elements including screens and processing logic for each screen, the original design-time representation including one or more application screens and original processing logic for each application screen,

the original processing logic including a call to a run-time module in the first run-time

receiving an original design-time representation of an application, the original design-time representation for use in a first run-time environment for executing applications having been developed in a first design-time environment, the first design-time environment using a first programming model comprising one or more first model elements including screens and processing logic for each screen, the original design-time representation including one or more application screens and original processing logic for each application screen; and

generating a converted design-time representation of the application based on the original design-time representation,

wherein the original processing logic comprises state control logic and one or more

environment; and

the converted design-time representation of the application is for use in a second run-time environment for executing applications having been developed in a second design-time environment, the second design-time environment using a second programming model comprising one or more second model elements including models, views, and controllers, the converted design-time representation including one or more application views based on the one or more application screens, and converted processing logic based on the original processing logic, the converted processing logic being capable of being executed in the second run-time environment; and

executing the run-time code in the second run-time environment using an adapter operable to interface with the run-time module in the first run-time environment.

calls to one ore more run-time modules in the first run-time environment,

the converted design-time representation of the application for use in a second run-time environment for executing applications having been developed in a second design-time environment, the second design-time environment using a second programming model comprising one or more second model elements including models, views, and controllers, the converted design-time representation including one or more application views based on the one or more application screens, and converted processing logic based on the original processing logic, the converted processing logic being capable of being executed in the second run-time environment.

generating corresponding state control logic that is executable by an adapter in the second run-time environment, the adapter being operable to interface with the run-time

modules in the first run-time environment;

As illustrated above, claim 6 of the copending application recites "a converted design-time representation of the application," but does not expressly recite "receiving run-time code for an application, the run-time code being generated" from the converted design-time representation of the application. Similarly, claim 6 of the copending application does not expressly recite the operation of "executing the run-time code in the second run-time environment" as recited in claim 1 of the present application.

However, claim 6 of the copending application does recite, for example, that "the converted processing logic" is "capable of being executed in the second run-time environment," and that "converting the original processing logic" comprises "converting the calls to the run-time modules into instructions to the adapter for invoking the run-time modules." In other words, claim 6 of the copending application suggests, to one of ordinary skill in the art, executing the application in the second run-time environment and invoking run-time modules. Similarly, the execution of the application in a run-time environment implies, to one of ordinary skill in the art, the existence of run-time code for the application.

Therefore, generating run-time code for the application from the converted design-time representation, such as with a compiler or some other code generator, and subsequently executing the run-time code, would have been obvious to one of ordinary skill in the art. Accordingly, claim 1 of the present application would have been obvious over claim 6 of the copending application. Similar reasoning applies to the other conflicting claims.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

Claim Rejections - 35 USC § 101

5. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

6. Claims 1-11 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

With respect to claims 1-7, and similarly with respect to claims 8-11, the claims are directed to "a computer program product, tangibly embodied in an information carrier." However, Applicant's definition of an information carrier includes "a propagated signal" (specification, page 11, line 20). Such signals do not fall within any category of statutory subject matter. See *Interim Guidelines for Examination of Patent Applications for Patent Subject Matter Eligibility* (1300 OG 142), Annex IV, part (c). Accordingly, the claims are not limited to statutory subject matter. It is suggested that Applicant amend the claims to replace "an information carrier" with --a machine-readable storage device--, which is also included in Applicant's definition of an information carrier (specification, page 11, line 20).

Claim Rejections - 35 USC § 102

7. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this

subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

8. Claims 1, 3-8, 10-12, 14-16, 18 and 19 are rejected under 35 U.S.C. 102(e) as being anticipated by U.S. Patent No. 7,007,278 to Gungabeesoon ("Gungabeesoon").

With respect to claim 1, Gungabeesoon discloses a computer program product, tangibly embodied in an information carrier, the computer program product being operable to cause data processing apparatus to perform operations (see, for example, column 7, lines 2-19) comprising:

receiving run-time code for an application (see, for example, column 11, lines 2-7, which shows receiving run-time code for an application in the form of a JavaServer Page, and column 10, lines 42-49, which shows that the run-time code is for a legacy application), the run-time code being generated from a converted design-time representation of the application (see, for example, column 9, lines 14-17, which shows that the run-time code is generated from converted design-time representations of the application in the form of user interface pages), wherein:

the converted design-time representation of the application is generated from an original design-time representation of the application developed for use in a first run-time environment for executing applications having been developed in a first design-time environment (see, for example, column 8, lines 44-54, which shows that the converted design-time representations are generated from original design-time representations of the application in the form of screen definitions, and column 7, lines 50-56, which shows a first run-time environment for executing applications developed in a first design-time environment), the first design-time environment using a first programming model comprising one or more first model elements including screens and processing logic for each screen (see, for example, column 8, lines 12-29, which shows that

the first design-time environment uses a programming model comprising screens and processing logic), the original design-time representation including one or more application screens and original processing logic for each application screen (see, for example, column 8, lines 44-47, which shows that the original design-time representation includes one or more application screens), the original processing logic including a call to a run-time module in the first run-time environment (see, for example, column 8, lines 29-32, which shows that the original processing logic includes calls to run-time modules in the first run-time environment); and

the converted design-time representation of the application is for use in a second run-time environment for executing applications having been developed in a second design-time environment (see, for example, column 9, lines 41-48, which shows a second run-time environment for executing applications developed in a second design-time environment), the second design-time environment using a second programming model comprising one or more second model elements including models, views, and controllers (see, for example, column 8, line 67 to column 9, line 7, which shows that the second design-time environment uses a programming model that comprises views in the form of HTML code and controllers in the form of JavaScript code, and column 9, line 54 to column 10, line 2, which shows that the programming model comprises models in the form of JavaBean data objects), the converted design-time representation including one or more application views based on the one or more application screens, and converted processing logic based on the original processing logic, the converted processing logic being capable of being executed in the second run-time environment (see, for example, column 10, lines 31-36, which shows that the converted design-time representation includes one or more application views and processing logic based on the original

application screens and processing logic, and column 10, lines 42-49, which shows that the converted processing logic is executable in the second run-time environment); and

executing the run-time code in the second run-time environment using an adapter operable to interface with the run-time module in the first run-time environment (see, for example, column 11, lines 8-22, which shows executing the run-time code in the second run-time environment and interfacing with the first run-time environment, and column 10, lines 3-17, which shows that the interface is an adapter in the form of a Publish-to-Web component).

With respect to claim 3, the rejection of claim 1 is incorporated, and Gungabeesoon further discloses that executing the run-time code comprises using the adapter to perform a function not performed by the original processing logic (see, for example, column 9, lines 17-27, which shows that executing the run-time code comprises using the adapter to perform functions that the original processing logic does not perform).

With respect to claim 4, the rejection of claim 3 is incorporated, and Gungabeesoon further discloses that the function comprises input validation (see, for example, column 8, line 67 to column 9, line 7, which shows performing input validation).

With respect to claim 5, the rejection of claim 3 is incorporated, and Gungabeesoon further discloses that the function comprises input formatting (see, for example, column 9, lines 28-31, which shows performing input formatting).

With respect to claim 6, the rejection of claim 1 is incorporated, and Gungabeesoon further discloses that:

the original design-time representation of the application comprises original state control logic (see, for example, column 7, lines 60-66, which shows that the application comprises original state control logic; and

the converted design-time representation of the application comprises converted state control logic based on the original state control logic, the converted state control logic capable of being executed by the adapter (see, for example, column 8, lines 5-11, which shows that the converted design-time representation comprises converted state control logic, and column 11, lines 8-22, which shows that the adapter executes the converted state control logic)..

With respect to claim 7, the rejection of claim 1 is incorporated, and Gungabeesoon further discloses that:

the original design-time representation of the application comprises one or more controls from a first set of controls (see, for example, column 8, lines 54-57, which shows that the original design-time representation comprises one or more user interface elements or controls);

the converted design-time representation of the application comprises one or more controls from a second set of controls, each control in the converted design-time representation of the application corresponding to a control in the original design-time representation of the application (see, for example, column 9, lines 54-60, which shows that the converted design-time representation comprises one or more controls corresponding to the controls in the original design-time representation); and

executing the run-time code comprises rendering the controls in the converted design-time representation of the application (see, for example, column 8, lines 44-54, which shows rendering the controls)..

With respect to claim 8, Gungabeesoon discloses a computer program product, tangibly embodied in an information carrier, the computer program product being operable to cause data processing apparatus to perform operations (see, for example, column 7, lines 2-19) comprising:

receiving run-time code for an application (see, for example, column 11, lines 49-55, which shows run-time code for an application comprising network application data and legacy application data, and see, for example, column 10, lines 37-42, and column 11, lines 2-7, which shows receiving such run-time code from a servlet instance);

determining whether the run-time code was generated from a native design-time representation of the application or from a converted design-time representation of the application (see, for example, column 10, lines 42-49, and column 11, lines 8-22, which shows the servlet instance responding in different ways depending on whether the run-time code comprises network application data or legacy application data, respectively).

Here, run-time code comprising network application data is considered run-time code that was generated from a native design-time representation of the application, and run-time code comprising legacy application data is considered run-time code that was generated from a converted design-time representation of the application. The servlet instance or some other related component in Gungabeesoon inherently determines whether the run-time code was generated from a native design-time representation of the application or from a converted design-time representation of the application, so as to respond in the appropriate manner. The native and converted design-time representations are further addressed below.

Gungabeesoon further discloses that:

the native design-time representation of the application is for use in a first run-time environment for executing applications having been developed in a first design-time environment (see, for example, column 9, lines 41-48, which shows a first run-time environment for executing applications developed in a first design-time environment), the first design-time environment using a first programming model comprising one or more first model elements including models, views, and controllers (see, for example, column 8, line 67 to column 9, line 7, which shows that the first design-time environment uses a programming model that comprises views in the form of HTML code and controllers in the form of JavaScript code, and column 9, line 54 to column 10, line 2, which shows that the programming model comprises models in the form of JavaBean data objects); and

the converted design-time representation of the application is generated from an original design-time representation of the application developed for use in a second run-time environment for executing applications having been developed in a second design-time environment (see, for example, column 8, lines 44-54, which shows that the converted design-time representations are generated from original design-time representations of the application in the form of screen definitions, and column 7, lines 50-56, which shows a second run-time environment for executing applications developed in a second design-time environment), the second design-time environment using a second programming model comprising one or more second model elements including screens and processing logic for each screen (see, for example, column 8, lines 12-29, which shows that the second design-time environment uses a programming model comprising screens and processing logic), the original design-time representation including one or more application screens and original processing logic for each application screen (see, for

example, column 8, lines 44-47, which shows that the original design-time representation includes one or more application screens), the converted design-time representation including one or more application views based on the one or more application screens, and converted processing logic based on the original processing logic, the converted processing logic capable of being executed in the second run-time environment (see, for example, column 10, lines 31-36, which shows that the converted design-time representation includes one or more application views and processing logic based on the original application screens and processing logic, and column 10, lines 42-49, which shows that the converted processing logic is executable in the second run-time environment); and

if the run-time code was generated from the native design-time representation, execute the run-time code in the first run-time environment using a set of run-time modules in the first run-time environment (see, for example, column 10, lines 42-49, which shows executing the run-time code generated from the native design-time representation in the first run-time environment using run-time modules such as the servlet instance in the first run-time environment); and

if the run-time code was generated from the converted design-time representation, execute the run-time code in the first run-time environment using a set of run-time modules in the second run-time environment (see, for example, column 11, lines 8-22, which shows executing the run-time code generated from the converted design-time representation in the first run-time environment using run-time modules such as the legacy application in the second run-time environment).

With respect to claim 10, the rejection of claim 8 is incorporated, and Gungabeesoon further discloses that executing the run-time code using the set of run-time modules in the second

run-time environment comprises using an adapter in the first run-time environment to interface with the set of run-time modules in the second run-time environment (see, for example, column 10, lines 3-17, which shows interfacing with the second run-time environment using an adapter in the form of a Publish-to-Web component in the first run-time environment).

With respect to claim 11, the rejection of claim 8 is incorporated, and Gungabeesoon further discloses that:

executing the run-time code using the set of run-time modules in the first run-time environment comprises using a first sequence of process steps (see, for example, column 10, lines 42-49, which shows a first sequence of process steps that comprises, for example, creating a socket and spawning a thread); and

executing the run-time code using the set of run-time modules in the second run-time environment comprises using a second sequence of process steps (see, for example, column 11, lines 8-22, which shows a second sequence of process steps that comprises, for example, forwarding data to the socket).

With respect to claims 12, 14 and 15, the claims are directed to an apparatus that corresponds to the computer program product of claims 1, 6 and 3, respectively (see the rejection of claims 1, 6 and 3 above).

With respect to claims 16, 18 and 19, the claims are directed to a method that corresponds to the computer program product of claims 1, 6 and 3, respectively (see the rejection of claims 1, 6 and 3 above).

Claim Rejections - 35 USC § 103

9. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

10. Claims 2, 9, 13 and 17 are rejected under 35 U.S.C. 103(a) as being unpatentable over Gungabeesoon, as applied to claims 1, 8, 12 and 16 above, respectively, in view of "Database Performance in the Real World: TPC-D and SAP R/3" by Doppelhammer et al. ("Doppelhammer").

With respect to claim 2, the rejection of claim 1 is incorporated. Gungabeesoon discloses that the first programming model is a legacy programming model and the second programming model is a Web-based programming model (see, for example, column 8, lines 5-11), but does not expressly disclose that the first programming model is the SAP Dynpro programming model and the second programming model is the SAP Web Dynpro programming model.

However, Doppelhammer teaches the SAP Dynpro programming model comprising one or more model elements including screens and processing logic for each screen (see, for example, page 125, section 2.3, first paragraph).

Moreover, Gungabeesoon suggests that such programming models are supported (see, for example, column 11, lines 23-27), and further discloses that converting applications from the legacy programming model to the Web-based programming model enables one to access the

applications across the network and take advantage of Web-based technology without having to make code changes to the applications (see, for example, column 11, lines 42-55).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to apply the teachings of Gungabeesoon in cases where the first programming model is the SAP Dynpro programming model and the second programming model is the SAP Web Dynpro programming model, so as to take advantage of Web-based technology without having to make code changes to the applications.

With respect to claim 9, the rejection of claim 8 is incorporated. Gungabeesoon discloses that the first programming model is a Web-based programming model and the second programming model is a legacy programming model (see, for example, column 8, lines 5-11), but does not expressly disclose that the first programming model is the SAP Web Dynpro programming model and the second programming model is the SAP Dynpro programming model.

However, Doppelhammer teaches the SAP Dynpro programming model comprising one or more model elements including screens and processing logic for each screen (see, for example, page 125, section 2.3, first paragraph).

Moreover, Gungabeesoon suggests that such programming models are supported (see, for example, column 11, lines 23-27), and further discloses that converting applications from the legacy programming model to the Web-based programming model enables one to access the applications across the network and take advantage of Web-based technology without having to make code changes to the applications (see, for example, column 11, lines 42-55).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to apply the teachings of Gungabeesoon in cases where the first programming model is the SAP Dynpro programming model and the second programming model is the SAP Web Dynpro programming model, so as to take advantage of Web-based technology without having to make code changes to the applications.

With respect to claim 13, the claim is directed to an apparatus that corresponds to the computer program product of claim 2 (see the rejection of claim 2 above).

With respect to claim 17, the claim is directed to a method that corresponds to the computer program product of claim 2 (see the rejection of claim 2 above).

Conclusion

11. The prior art made of record and not relied upon is considered pertinent to Applicant's disclosure (see the attached Notice of References Cited).

12. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Yigdall whose telephone number is (571) 272-3707. The examiner can normally be reached on Monday through Friday from 7:30am to 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

MY

Michael J. Yigdall
Examiner
Art Unit 2192

mjy

TUAN DAM
SUPERVISORY PATENT EXAMINER